Universitatea Tehnică „Gheorghe Asachi” din Iași

Facultatea de Automatică și Calculatoare

Domeniul: Tehnologia Informației

Shopping Application

“PBD Express”

Proiect la disciplina

Proiectarea Bazelor de Date

Studenți: Enachi Vasile

Strilciuc Gabriel

Grupa: 1409B

Coordonator: Sorin Avram

Capitolul 1. Introducere

Proiectul dat imită funcționalitatea unei baze de date în sfera shopping-ului online prin intermediul unui API (application programming interface).

API-ul din proiectul curent oferă utilizatorului posibilitatea de a naviga în tabela produselor, magazinelor și distribuitorilor. Pentru început user-ul este redirecționat pe pagina de 'login', unde acesta se poate autentifica cu profilul pe care îl deține sau poate creea un profil nou.

În aplicație există 4 tipuri de utilizatori, în funcție de tipul account-ului, fiecare utilizator dispune de roluri și privilegii diferite:

- client
- admin shop
- admin shipping
- global admin

Client

Utilizatorul cu tipul de account *client* beneficiază de posibilitatea de a căuta in tabelul de magazine, produse și distribuitori având la dispoziție mai multe field-uri de căutare în funcție de necesitate (nume, preț, preț range, id, locație ș.a.).

Admin Shop

Pe lângă avantajele pe care le are *clientul*, *administratorul de magazine* dispune de posibilitatea de a insera, modifica sau șterge înregistrări din tabelul **Shops** și tabelul **Products**.

Admin Shipping

Utilizatorul cu tipul account-ului *administrator de distribuitori* dispune de avantajele care le are *clientul*, precum și posibilitatea de inserare, modificare și stergere a înregistrărilor din tabelul **Shipping_Methods**.

Global Admin

Utilizatorul cu account-ul de tip *admin global* dispune de avantajele pe care le are atât *admin_shop*-ul cât și *admin_shipping*-ul. Pe lângă aceasta, dispune și de o pagină proprie de vizualizare a tuturor userilor, cu informațiile de rigoare (Nume, prenume, locație, email, phone, username, password) cu precizarea că acesta nu va putea vizualiza parolele celorlalți useri care au tipul de account *global_admin.*

Din scopuri pur didactice, asignarea tipurilor de account-uri se face la crearea account-ului userilor din interfața grafică.

Capitolul 2. Tehnologiile folosite pentru front-end și back-end

Baza de date folosită în această aplicație este SQLPlus. Pentru partea de front-end s-a folosit librăria tkinter din python.

Frame-ul principalul denumit container se află în clasa BdGui. Această clasă deține un dicționar cu alte 7 frame-uri, care în funcție de utilizator pot fi invocate prin intermediul funcției **tkraise()**. Relația dintre clasele LoginPage, SignUpPage, AdvancedAdminPage, HomePage, ShopPage, ShippingPage, ProductPage este de agregare față de clasa BdGui.

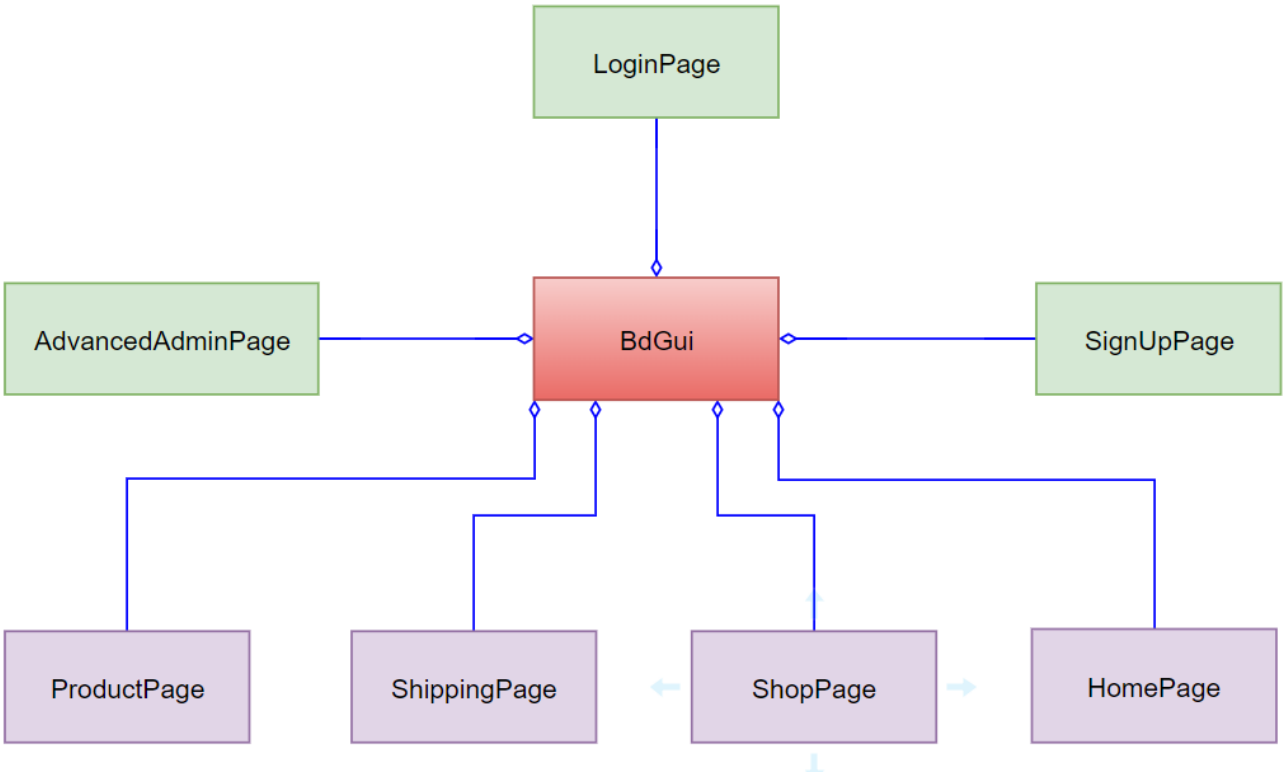

Fig 2.1. Diagrama de clase pentru paginile aplicației

Pentru vizualizarea înregistrărilor, a fost creată o clasă TableFrame care conține un atribut de tipul **tkinter.TreeView.** Clasele AdvancedAdminPage, HomePage, ShopPage, ShippingPage, ProductPage dețin ca atribut clasa TableFrame realizând o relație de compoziție.

Pentru introducerea datelor au fost folosite widget-urile tkinter.Entry, tkinter.Combobox și tkinter.CheckBox din cadrul librăriei tkinter. Restul diagramei de clase este prezentată în Fig. 2.2.

Criptarea şi decriptarea parolei

Pentru criptarea și decriptarea parolei sunt folosite metodele encrypt și decrypt din clasa Cipher, care utilizeaza clasa Fernet din librăria cryptography. Criptarea și decriptarea parolei este necesară atunci când baza de date nu rulează la nivel local, în cazul de față operațiile date sunt făcute în scop academic la logare și creare de account, înainte de executarea instrucțiunilor sql. Criptarea și decriptarea se face folosind o cheie, în cazul de față cheia fiind: "**my deep darkest secret is secure**" codificată într-un șir de octeți folosind librăria base64 (clasa Fernet accept doar chei '32 url-safe base64-encoded bytes').

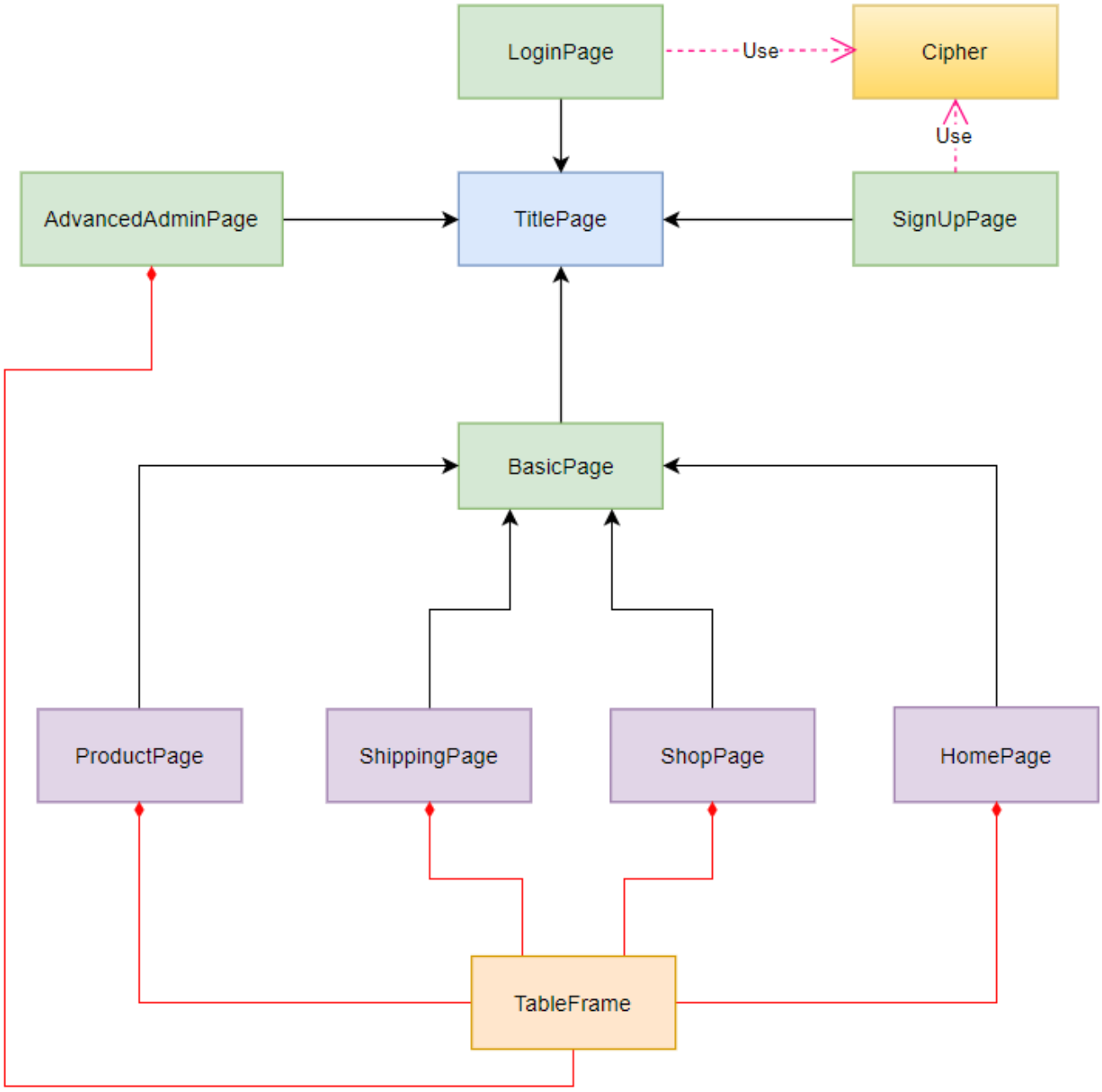

Fig 2.2. Diagrama de clase extinsă, fără clasa BdGui

Fig 2.3. Legendă

Capitolul 3. Structura și inter-relaționarea tabelelor

Structura tabelelor a fost creată așa încât să satisfacă unul din multiplele modele real a bazelor de date pentru shopping-ul real.

- Utilizatorilor le este permis să dețină o singură locație (la care va fi livrată comanda). Într-o locație pot exista mai mulți utilizatori (One-To-Many)
- Magazinelor le este permis să dețină o singură locație. Într-o locație pot exista mai multe magazine (One-To-Many)
- Într-un magazin pot exista mai multe produse, iar produsul înregistrat în baza de date aparține unui singur magazin (One-To-Many)
- O comandă poate conține doar un singur produs și un produs poate face parte din mai multe comenzi (One-To-Many). O comandă poate fi efectuată doar de un singur utilizator și un utilizator poate avea mai multe comenzi (One-To-Many). Acest lucru permite să se stabilească indirect o relație Many-To-Many între tabelul Products și App_Users (un utilizator poate comanda de mai multe ori același produs și același produs poate fi comandat de mai mulți utilizatori diferiți)
- O comandă poate fi livrată prin intermediul unui singur furnizor de livrări
- Un utilizator îi este asignat un singur account și un account poate fi deținut de un singur utilizator (relație One-To-One)

Ținând cont de precizările de mai sus, Diagrama ER (Entity-Relation) construită după modelul Crow's Foot Notation este reprezentată în figura Fig. 3.1.

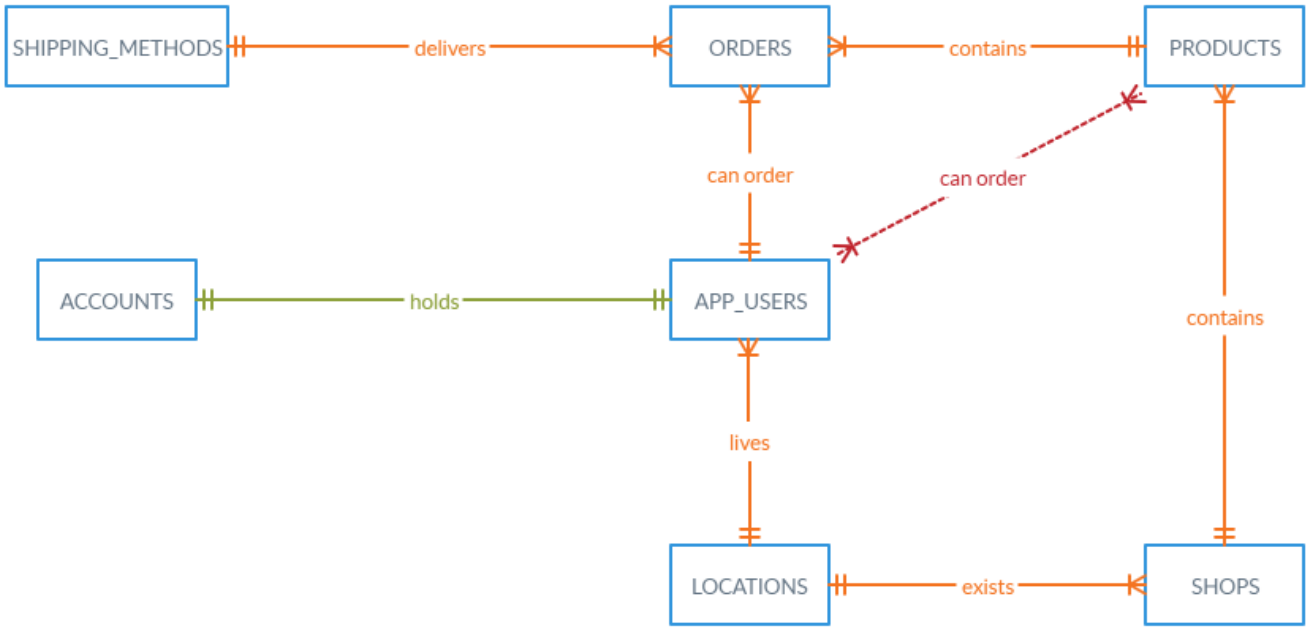

Fig. 3.1 Diagrama ER (Crow's Foot Notation)

Capitolul 4. Descrierea constrângerilor folosite

Constrângerile, atributele și cheile tabelelor sunt reprezentate în figura Fig.4.1.

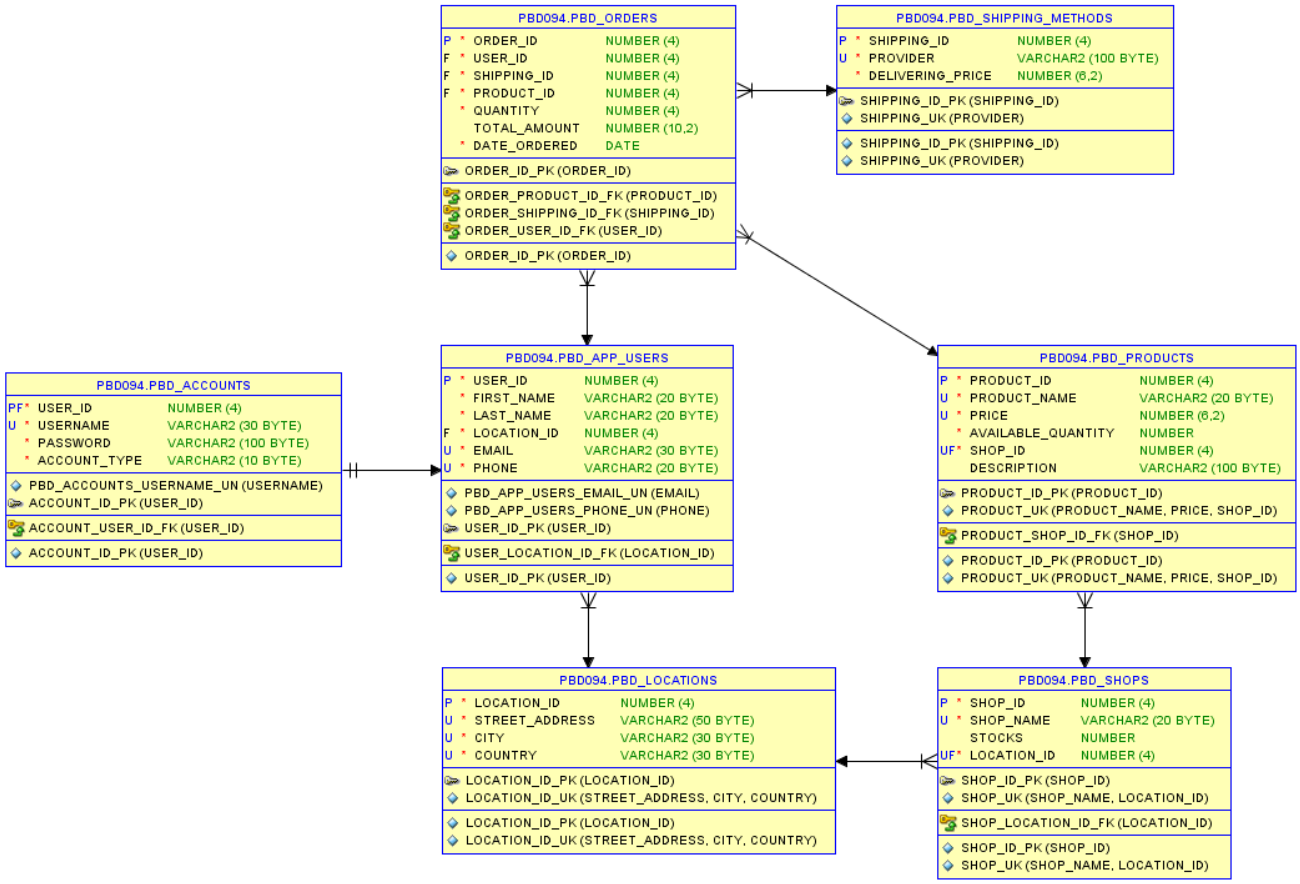

Fig. 4.1 Atributele, constrăngerile și cheile tabelelor

Pentru toate atributele din fiecare tabel, cu excepția atributului **Description** din tabelul **Products** s-a impus constrângerea de tip check – Not Null.

- **Shipping_Methods**

Primary Key: shipping_id
Constrângere Unique: provider (un furnizor are un nume unic)
Constrângere Check: delivering_price (prețul trebuie să fie pozitiv)

- **Orders**

Primary Key: order_id
Constrângere Check: quantity, total_amount (cantitatea și suma totală trebuie să fie pozitive)
Foreign Keys: user_id (app_users), shipping_id (shipping_methods), product_id (products)

- **Locations**

Primary Key: location_id
Constrângere Unique: (street_address, city, country)

- **Shops**

Primary Key: shop_id
Constrângere Unique: (shop_name, location_id)
Foreign Keys: location_id

- **Products**

Primary Key: product_id
Constrângere Unique: (product_name, price, shop_id)
Constrângere Check: price (prețul trebuie să fie pozitiv)
Foreign Keys: shop_id

- **Accounts**

Primary Key: user_id
Constrângere Check: password (parola trebuie să aibă minim 6 caractere)
Constrângere Unique: username (un account trebuie să aibă un nume de utilizator unic)
Foreign Keys: user_id

- **App_Users**

Primary Key: user_id
Constrângere Unique: email, phone
Foreign Keys: location_id

Normalizare

Toate tabelele sunt aduse cel puțin la a 3 formă normală.

- Un atribut conține valori atomice din domeniul său și nu grupuri de valori (I formă)
- Atributele non-cheie depind de toate cheile candidat (a II-a formă)
- Atributele non-cheie nu sunt tranzitiv dependente de cheile candidat (a III-a formă)

Forma normală Boyce-Codd este de asemenea asigurată, întrucât:

- Pentru o dependență X→Y, resultă că X – super cheie

Descrierea trigger-ilor

Primary Key-urile de tip id sunt generate de baza de date după tipul AUTO-Increment. Pentru acest lucru se crează o secvență de increment, care se folosește în triggere de inserare.

Alți triggeri folosiți în dezvoltarea aplicaţiei sunt folosite la partea de validare a ştergerii unei înregistrări care se află într-o relaţie de tip foreign key. Astfel de triggere sunt folosite pe tabelele shipping_methods şi products.

De asemenea, la achiziționarea unui product (adăugare in tabelul *pbd_orders*), are loc modificare field-ului *total_amount* printr-un trigger de tip *trigger row before*. Apoi, printr-un al trigger de tip *trigger row after*, are loc actualizarea cantității valabile în cadrul produselor și adăugarea de fund-uri în tabelul *pbd_shops.*

Un alt trigger este orientat pe tabelul *pbd_products*, unde adăugarea unui produs depinde de stoc-urile pe care le are magazinul (eroare la insuficiență de stoc-uri), iar ștergerea produsului adaugă stocuri în magazin.

Descrierea pachetelor

S-au creat pachete pentru operațiile de tip delete, insert și update pentru entitățile *pbd_orders, pbd_shipping_methods, pbd_products* și *pbd_shops.*

De asemenea s-a creat un pachet utils, care este folosit pentru afișarea în cadrul tranzacțiilor create.

Descrierea excepțiilor

S-au creat excepții de tip *Raise_Application_Error* pentru tranzacțiile descrise mai sus, care sunt folosite pentru a afișa mesaj de eroare în cadrul interfeței GUI din Python.

Capitolul 5. Descrierea modalității de conectare la baza de date

Conectarea la baza de date a fost făcută prin intermediul librăriei cx_Oracle.

```
def run_query(self, query):
    cursor = self.conn.cursor()
    cursor.execute(query)
    try:
        query_results = [row for row in cursor]
        self.conn.commit()
        cursor.close()
    except cx_Oracle.InterfaceError:
        self.conn.commit()
        cursor.close()
        return None
    return query_results
```

Metoda run_query execută o interogare SQL și returnează rezultatele într-o listă de tuple. Această metodă aparține clasei BdGui.

Informații auxiliare

În program sunt verificate datele introduse de utilizator conform constrângerilor impuse în baza de date. Nerespectarea datelor de interogare este atenționată utilizatorului prin intermediul unei ferestre de tip Pop-Up, folosind widgetul tkinter.messagebox.

În clasa TableFrame a fost definită o metodă de sortare a datelor din interfață crescător sau descrescător, la apăsarea coloanei cu mouse-ul de către utilizator.

Clasa TableFrame conține o metodă de sortare a datelor după coloana selectată.

Capitolul 6. Capturi de ecran

- **Interfața grafică**